Machine Learning for Constraint Acquisition

Luc De Raedt

Questions in ICON

1. Can CP problems and CP solvers help to formulate and solve ML / DM problems ?
2. Can ML and DM help to formulate and solve constraint satisfaction problems ?

We shall argue that the answer to both questions is YES

The CP perspective

Formulating the model is a knowledge acquisition task

Improving the performance of solvers is speed-up learning

Machine learning may help as shown by several initial works

The ML/DM Perspective

Machine Learning is a (constrained) optimization problem

- learning functions

Data mining is often constraint satisfaction

- “Constraint based mining”

Still ML/DM do not really use CP ...

How ML might help CP

Machine Learning for CP

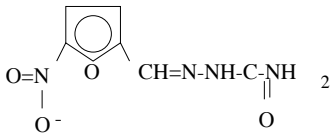
CSP (V,D,C,f) (f: Optimisation function)

At least three interpretations

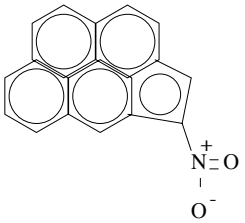
- Learning CSP(V,D,C,f) from examples
- Learning to solve for better performance
  - "clause" learning etc. (speed-up learning, explanation based learning)
  - learning portfolio's of solvers (meta-learning, preference learning)

Structure Activity Relationship Prediction

Active



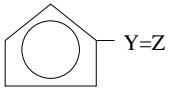
nitrofurazone



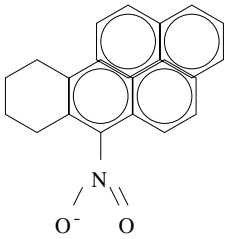
4-nitropenta[cd]pyrene

[Srinivasan et al. AIJ 96]

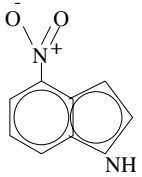
Structural alert:



Inactive



6-nitro-7,8,9,10-tetrahydrobenzo[a]pyrene



4-nitroindole

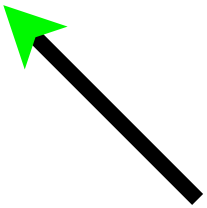
Data = Set of Small Graphs

Machine Learning

Given

- a space of possible instances X
- an unknown target function $f: X \rightarrow Y$
- a hypothesis space L containing functions $X \rightarrow Y$
- a set of examples $E = \{ (x, f(x)) \mid x \in X \}$
- a loss function *loss*$(h,E) \rightarrow \mathbb{R}$

Find $h \in L$ that minimizes *loss*(h,E)



supervised

Classification

Given - Molecular Data Sets

- a space of possible instances X -- Molecular Graphs
- an unknown target function f: $X \rightarrow Y$ -- {Active,Inactive}
- a hypothesis space L containing functions $X \rightarrow Y$ -- L= {Active iff structural alert s covers instance $x \in X | s \in X$ }
- a set of examples $E = \{ (x, f(x)) \mid x \in X \}$
- a loss function *loss*(h,E) $\rightarrow \mathbb{R}$ $|\{ x \in E \mid f(x) \neq h(x)\}|$

Find $h \in L$ that minimizes *loss*(h,E)

If classes = {positive, negative} then this is concept-learning

Regression

Given - Molecular Data Sets

- a space of possible instances X -- Molecular Graphs
- an unknown target function f: X → Y -- $\mathbb{R}$
- a hypothesis space L containing functions X → Y -- a linear function of some features
- a set of examples E = { (x, f(x)) | x ∈ X }
- a loss function *loss*(h,E) → $\mathbb{R}$

$$\sqrt{\sum_{x \in E} f(x)^2 - h(x)^2}$$

- **Find** h ∈ L that minimizes *loss*(h,E)

Learning Probabilistic Models

Given

- a space of possible instances X
- an unknown target function P: X → Y Y=[0,1]
- a hypothesis space L containing functions X → Y (graphical models)
- a set of examples E = { (x, _) | x ∈ X }
- a loss function *loss*(h,E) → $\mathbb{R}$

generative

$$\prod_{e \in E} P(e|h)$$

Find h ∈ L that minimizes *loss*(h,E)

maximize likelihood

generative

Basic Setting

Boolean Concept-Learning

$X = \{(X_1, ..., X_n) \mid X_i = 0 / 1\}$

$Y = \{+,-\}$

L = boolean formulae

loss(h,E) = training set error

$= |\{e \mid e \in E, h(e) \neq f(e)\}| / |E|$

sometimes required to be 0

Simplest setting for learning, compatible with DM part and with CP

Dimensions

Given

- a space of possible instances X
- an unknown target function $f: X \rightarrow Y$
- a hypothesis space L containing functions $X \rightarrow Y$ k-CNF ? DNF ? etc
- a set of examples $E = \{ (x, f(x)) \mid x \in X \}$ pos and neg ? or pos only
- a loss function *loss*(h,E) $\rightarrow \mathbb{R}$ loss/error=0 required ?

Find $h \in L$ that minimizes *loss*(h,E)

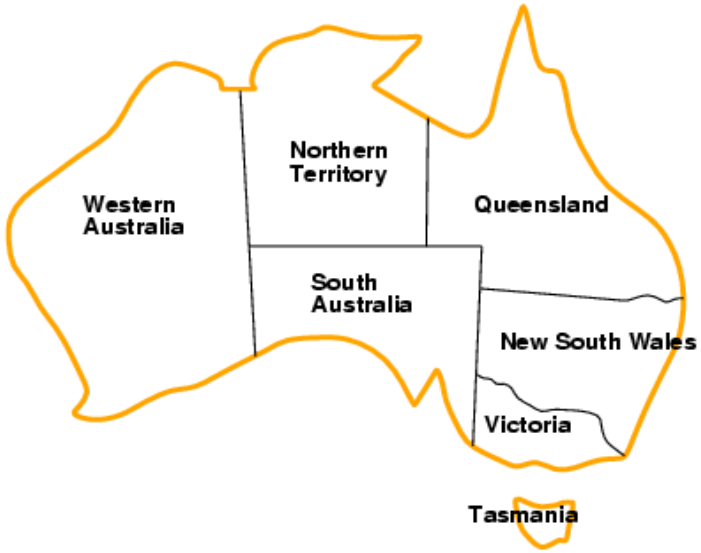
ability to ask questions ?

Boolean concept-learning

	1	2	3	4	5		
ex 1	0	1	0	1	0	..	+
ex 2	1	1	1	1	1		+
ex 3	0	1	1	0	0		-
ex 4	1	0	0	1	0		-
...							

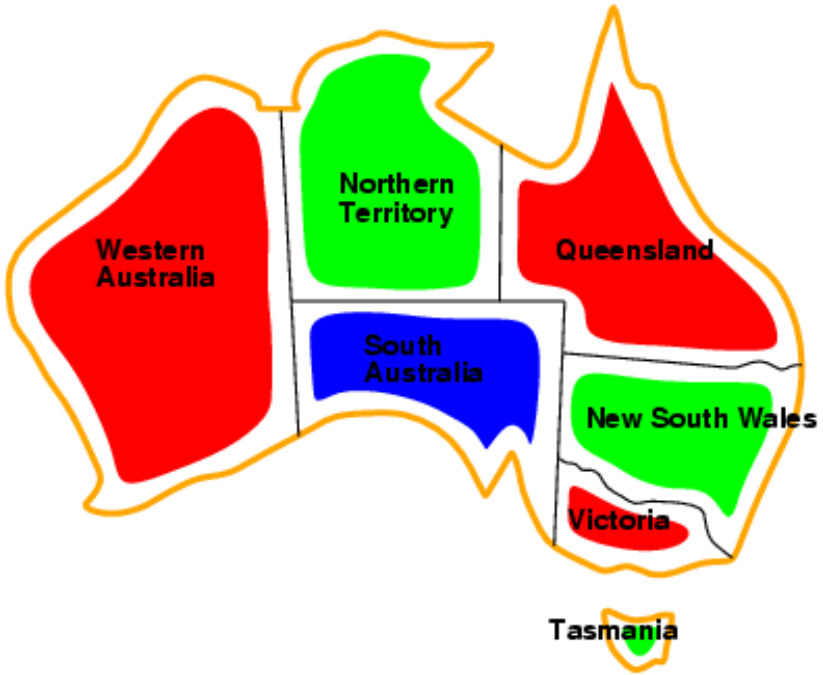
X_2 and X_4

Map Colouring



- Variables WA, NT, Q, NSW, V, SA, T
- Domains D_i = {red,green,blue}
- Constraints: adjacent regions must have different colors
- e.g., WA ≠ NT, or (WA,NT) in {(red,green),(red,blue),(green,red), (green,blue),

Map Colouring



- Solutions are complete and consistent assignments, e.g., WA = red, NT = green,Q = red,NSW = green,V = red, SA = blue,T = green

Why boolean concept-learning ? constraint networks

(V_1, V_2, V_3)	$V_1 < V_2$	$V_1 > V_2$	$V_1 = V_2$	$V_1 < V_3$	
(1,2,3)	1	0	0	1	
(2,3,1)	1	0	0	0	
(3,2,1)	0	1	0	0	
(1,3,2)	1	0	0	1	
...					

Bias

Propositionalization

CONACQ example *[Bessiere et al.]*

Monomials

Given

- a space of possible instances X
- an unknown target function $f: X \rightarrow Y$
- a hypothesis space L containing functions $X \rightarrow Y$ — monomials conjunctions
- a set of examples $E = \{ (x, f(x)) \mid x \in X \}$ — pos only
- a loss function *loss*(h,E) $\rightarrow \mathbb{R}$ — error = 0

Find $h \in L$ that minimizes *loss*(h,E)

Learning monomials

Represent each example by its set of literals

- $\{\neg X_1 , X_2 , \neg X_3 , X_4 , \neg X_5 \}$

Compute the intersection of all positive examples

- intersection = least general generalization

A cautious algorithm

Makes prudent generalizations

[Mitchell, ML textbook 97]

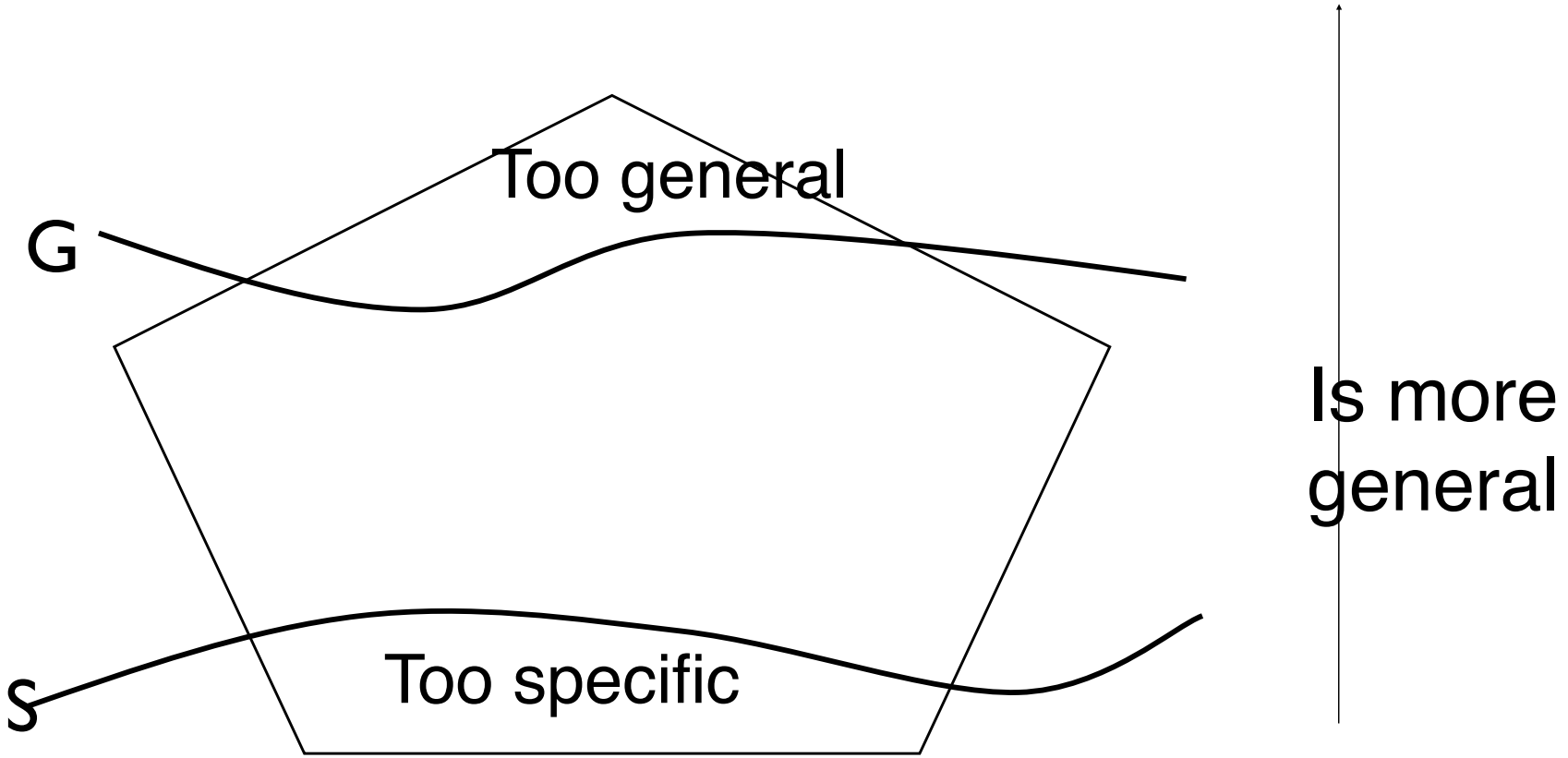
Find-S algorithm

Complaints about Find-S

- Can't tell whether it has learned concept
- Can't tell when training data inconsistent
- Picks a maximally specific h (why?)
- Depending on H, there might be several.

- Could be alleviated with Versionspace Approach

Mitchell's Versionspace



To remember

- Essential components of a learner
- Traversing / enumerating the pattern space L
  - use an operator (refinement operator)
  - naively (generate and test)
    - avoid generating the same pattern twice
  - pruning the space
  - heuristically (usually not for use in CP)
- stopping criterion and scoring criterion (using data)

Search

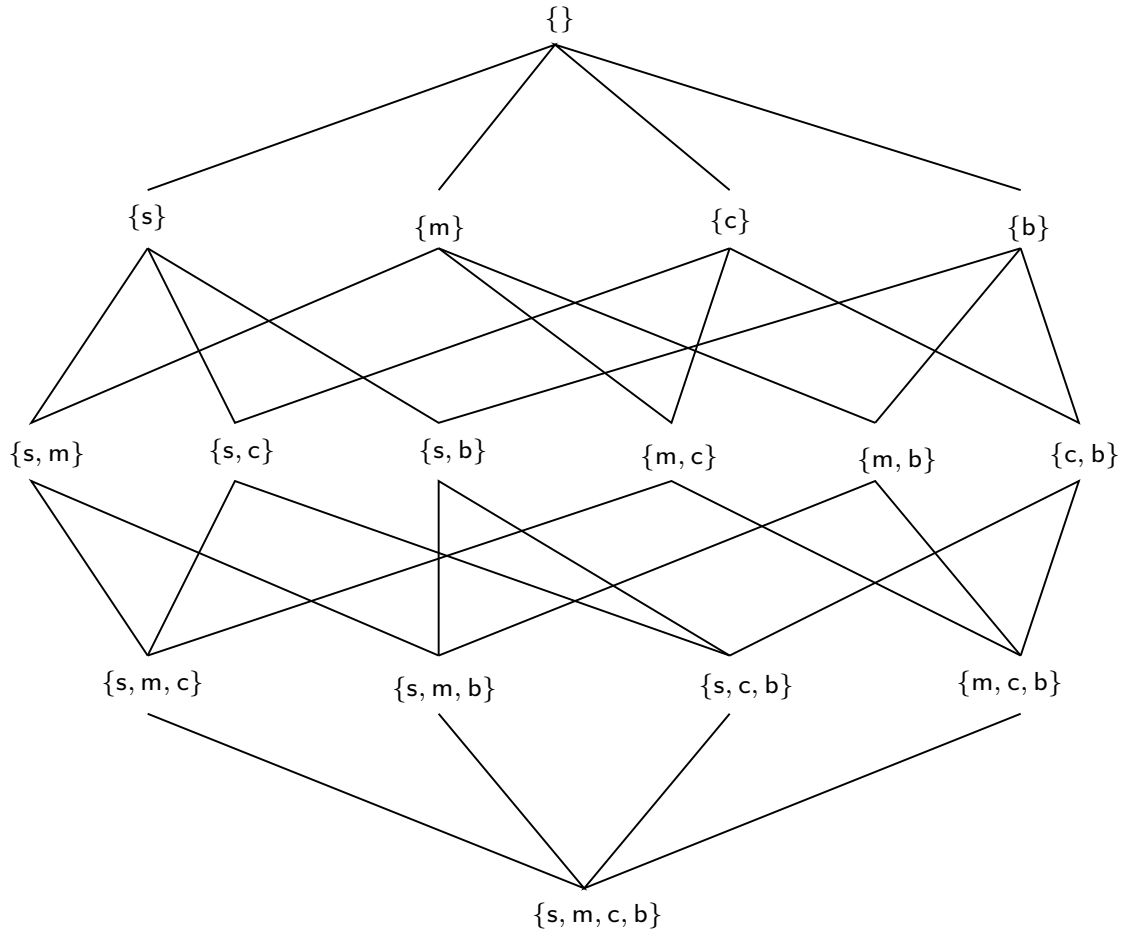


Fig. 3.5. The partial order over the item-sets

Monomials

Given

- a space of possible instances X
- an unknown target function $f: X \rightarrow Y$
- a hypothesis space L containing functions $X \rightarrow Y$ — monomials conjunctions
- a set of examples $E = \{ (x, f(x)) \mid x \in X \}$ — pos only
- a loss function *loss*(h,E) $\rightarrow \mathbb{R}$ — error = 0

Find $h \in L$ that minimizes *loss*(h,E)

Variations

Variations

- Asking questions
- Changing the representation
- Learnability

Asking Queries
Active Learning

Provide the learner with the opportunity to ask questions

Let T be the (unknown) target theory

- Does x satisfy T ? (membership)
- Does T |= X ? (subset)
- Does X |= T ? (superset)
- Are T and X logically equivalent ? (equivalence)
- ...

The oracle has to provide a counter-example in case the answer is negative *[Angluin, MLJournal 88]*

How can we use this?

Reconsider learning monomials (cf. *[Mitchell],* Conacq *[Bessiere et al]*)

Current hypothesis / conjunction

- $\{\neg X_1 , X_2 , \neg X_3 , X_4 , \neg X_5 \}$
- generate example $\{X_1 , X_2 , \neg X_3 , X_4 , \neg X_5 \}$
- if positive, delete X_1 , if negative, keep
- only n+1 questions needed to converge on unique solution (mistake bound)

Very interesting polynomial time algorithms for learning horn sentences *[Angluin et al. MLJ 92; Frazier and Pitt, ICML 93]* by asking queries

Quacq and Conacq

By Christian Bessiere et al.

They exploit these ideas in a systematic way

They also are interested in complexity

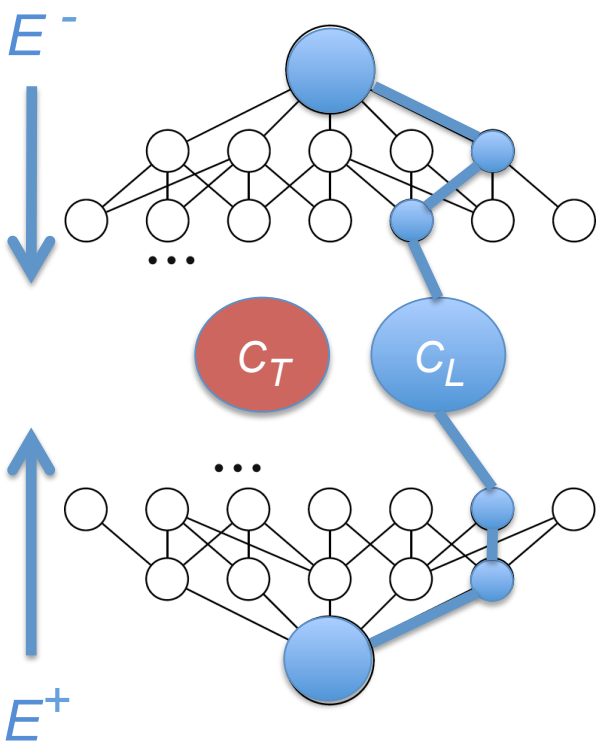
- how many questions needed ?
- optimal number of questions ?

Different types of questions.

Search bi-directional (idea of versionspaces)

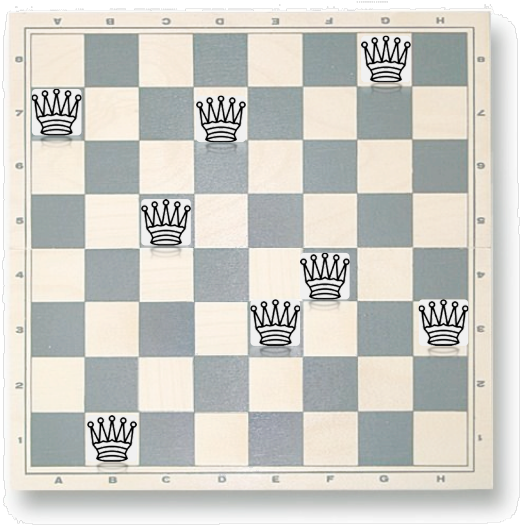
QuAcq: Quick Acquisition

- Bidirectional learning
- Based on ***partial*** queries
  - Positive example →prune the bias
  - Negative example → elucidate the scope of a constraint



Slide Bessiere et al.

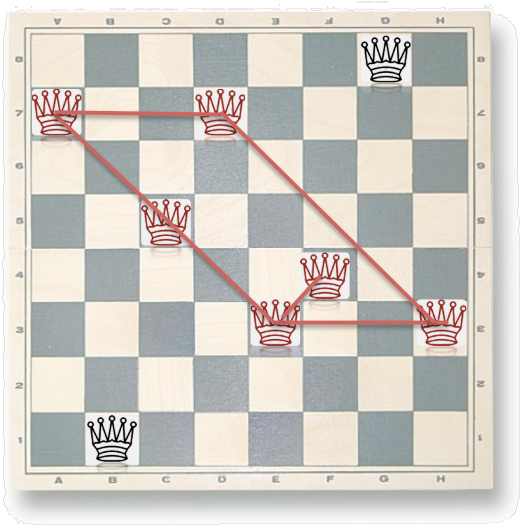
Partial Queries



ask(2, 8, 4, 2, 6, 5, 1, 6)

Slide Bessiere et al.

Partial Queries



ask(2, 8, 4, 2, 6, 5, 1, 6) = No

Slide Bessiere et al.

Types of queries

Membership :

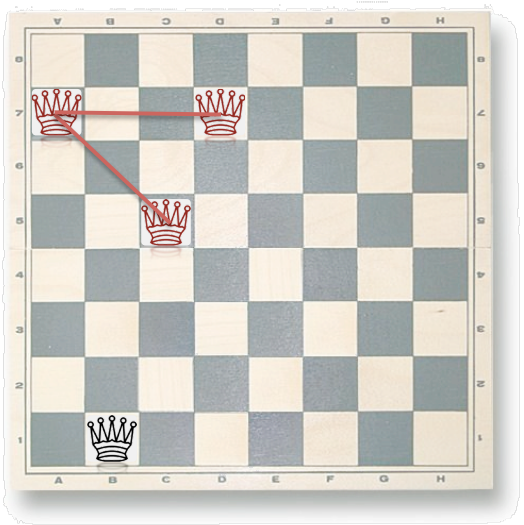
- does the board satisfy all constraints, ie is it a solution ? yes / no : positive negative examples

Partial queries

- does a partial board violate a constraint ? yes / no -> provides more information, faster convergence

Negative examples reduced using partial queries ...

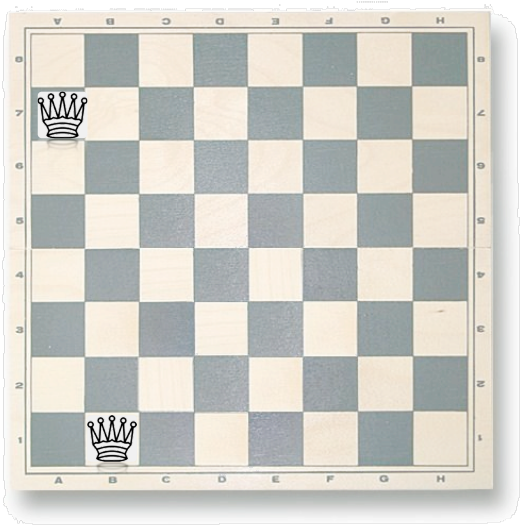
Partial Queries



ask(2, 8, 4, 2, -, -, -, -) = No

Slide Bessiere et al.

Partial Queries



ask(2, 8, -, -, -, -, -, -) = Yes

Slide Bessiere et al.

Complexity of QuAcq

- The number of queries required to find the target concept is in:

$$O(|C_T| \cdot (\log |X| + |\Gamma|))$$

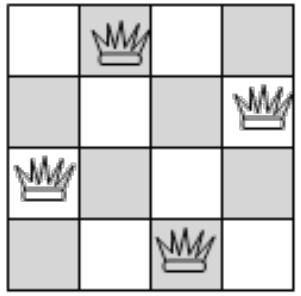
- The number of queries required to converge is in:

$$O(|B|)$$

Slide Bessiere et al.

Learn&Solve

- Example (4-queens)



e= 1 1 ✗ $C_L = \{q_1 \neq q_2\}$

Find(Fixed)Scope(e)

1 2 ✗ $C_L = C_L \cup \{q_1 \neq q_2 + 1\}$

1 3 ✓

extend 1 3 1 ✗ $C_L = C_L \cup \{q_1 \neq q_3\}$

2 4 1 ✓

extend 2 4 1 1 ✗ $C_L = C_L \cup \{q_3 \neq q_4\}$

2 4 1 3 ✓

slide Nadjib Lazaar

k-CNF

Given

- a space of possible instances X
- an unknown target function $f: X \rightarrow Y$
- a hypothesis space L containing functions $X \rightarrow Y$ k-CNF
- a set of examples $E = \{ (x, f(x)) \mid x \in X \}$ pos only
- a loss function *loss*(h,E) $\rightarrow \mathbb{R}$

Find $h \in L$ that minimizes *loss*(h,E)

Learning k-CNF

Naive Algorithm *[Valliant CACM 84]*

- Let S be the set of all clauses with k literals
- for each positive example e
  - for all clauses s in S
    - if e does not satisfy s then remove s from S

polynomial (for fixed k) -- PAC-learnable

Example

Suppose k = 3 and three variables A, B, C and

target = A v B v C and not A v not B v not C

Initial Theory

A v B v C

not A v B v C

A v not B v C

A v B v not C

not A v not B v C

A v not B v not C

not A v B v not C

not A v not B v not C

Example

Suppose k = 3 and three variables A, B, C and

target = A v B v C and not A v not B v not C

Example A & B & not C

A v B v C

not A v B v C

A v not B v C

A v B v not C

not A v not B v C

A v not B v not C

not A v B v not C

not A v not B v not C

Example

Suppose k = 3 and three variables A, B, C and

target = A v B v C and not A v not B v not C

Example A & B & not C

A v B v C

not A v B v C

A v not B v C

A v B v not C

~~not A v not B v C~~

A v not B v not C

not A v B v not C

not A v not B v not C

Example

Suppose k = 3 and three variables A, B, C and

target = A v B v C and not A v not B v not C

Example A & B & C

A v B v C

not A v B v C

A v not B v C

A v B v not C

~~not A v not B v C~~

A v not B v not C

not A v B v not C

~~not A v not B v not C~~

Learning (k)-CNF

Alternative algorithm using Item-Set Mining principles

- minimum frequency = 100%
- clauses are disjunctions; itemsets conjunctions
- monotonicity property :
  - if e satisfies clause C then e also satisfies C U { lit }
  - interest in smallest clauses that satisfy 100% freq.
- frequency({ }) = 0, so refinement needed as for item-sets
- find upper border ...

Search

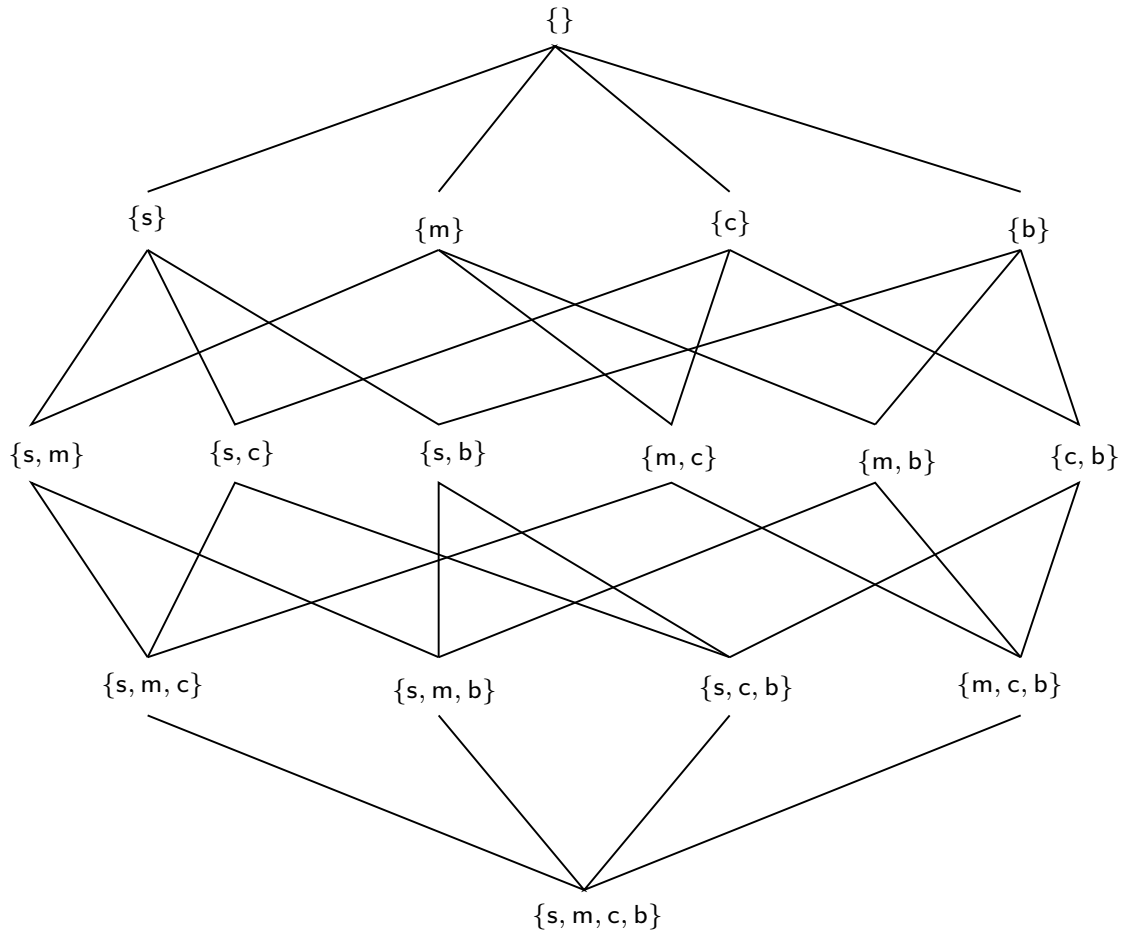


Fig. 3.5. The partial order over the item-sets

Where do the examples come from ?

Unkown probability distribution *P* is assumed on X

The examples in E are drawn at random according to *P*

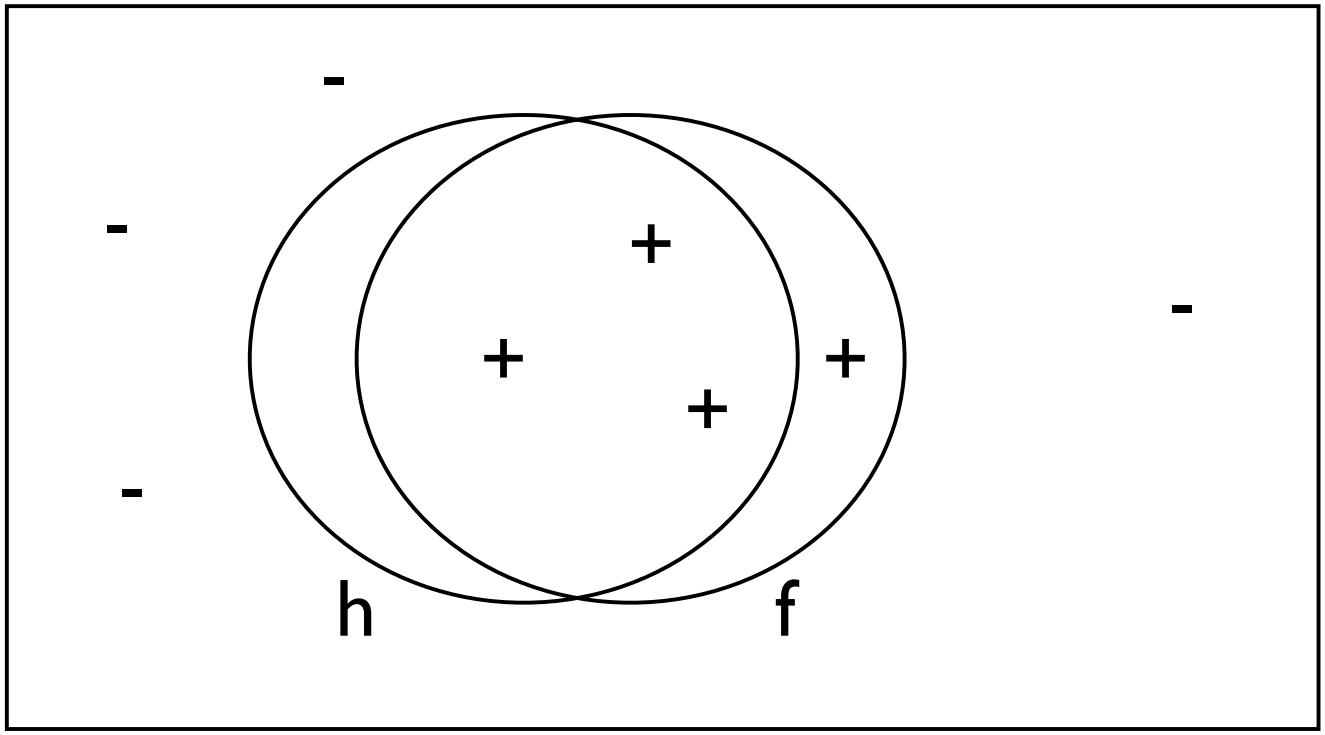
The i.i.d. assumption:

identically and independently distributed

(often does not hold for network / relational data)

Interpretation

Probability Distribution *P*



Classification Revisited

Make predictions about *unseen* data

$loss_l(h,E) = |\{e \mid e \in E, h(e) \neq f(e)\}| \; / \; |E|$

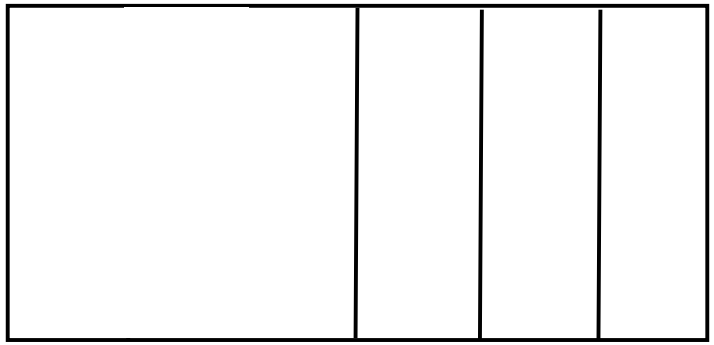
$= \text{training set error}$

$loss_t(h,X) = P(\{e \mid e \in X, h(e) \neq f(e)\})$

$= \text{true error}$

Estimating true error

Cross-validation



E_j

- Split E into E_1 ... E_k
- Repeat k times
  - Learn on k-1 sets E_i
  - Compute training error on left out E_j
  - Average

Also used for comparing and evaluating algorithms

Formal Frameworks Exist

Probably Approximately Correct learning (PAC)

requires that learner finds with high probability approximately correct hypotheses

So, $P(loss_t(h,X) < \varepsilon) > 1-\delta$

Typically combined with complexity requirements

- sample complexity: number of examples
- computational complexity

Valliant proved polynomial PAC-learnability (fixed k)

DNF / rule learning

Given

- a space of possible instances X
- an unknown target function f: X → Y
- a hypothesis space L containing functions X → Y DNF
- a set of examples E = { (x, f(x)) | x ∈ X } pos pos and neg
- a loss function *loss*(h,E) → $\mathbb{R}$ error need not be 0

Find h ∈ L that minimizes *loss*(h,E)

Rule learning

Learning from Positives and Negatives

Learn a formula in Disjunctive Normal Form

Rule learning algorithms (machine learning)

Similar issues to pattern set mining (data mining perspective)

Rule learning is often heuristic

Set-covering algorithm

- repeatedly search for one rule (conjunction) that covers many positives and no negative
- discard covered positive examples and repeat

[Fuernkranz, AI Review 99, book 2010/11]

Modelseker

By N. Beldiceanu and H. Simonis

The state-of-the-art in learning global constraint models

Motivating Example

2	-1	4	-3	6	-5	8	-7	10	-9	12	-11	14	-13	16	-15	18	-17
-8	15	-10	7	-18	9	-4	1	-6	3	-14	13	-12	11	-2	17	-16	5
4	-17	14	-1	12	-13	10	-15	16	-7	18	-5	6	-3	8	-9	2	-11
7	11	-8	15	-16	-18	-1	3	-12	13	-2	9	-10	17	-4	5	-14	6
-3	-13	1	-11	10	16	-15	-17	14	-5	4	-18	2	-9	7	-6	8	12
15	9	-7	13	-14	-12	3	11	-2	17	-8	6	-4	5	-1	18	-10	-16
-17	-5	-15	-18	2	14	-9	-13	7	-11	10	-16	8	-6	3	12	1	4
11	18	5	9	-3	-10	17	12	-4	6	-1	-8	-15	16	13	-14	-7	-2
-13	-6	-17	-5	4	2	-11	-9	8	-16	7	14	1	-12	-18	10	3	15
9	16	11	6	-8	-4	13	5	-1	12	-3	-10	-7	18	17	-2	-15	-14
-5	-12	-13	-16	1	7	-6	-18	15	-14	17	2	3	10	-9	4	-11	8
6	14	9	12	-7	-1	5	16	-3	18	-15	-4	-17	-2	11	-8	13	-10
-18	-10	-12	-14	15	8	-16	-6	17	2	13	3	-11	4	-5	7	-9	1
12	-7	18	10	-17	-15	2	14	-11	-4	9	-1	16	-8	6	-13	5	-3
-14	4	-16	-2	11	17	-18	-10	13	8	-5	15	-9	1	-12	3	-6	7
10	-8	6	-17	-9	-3	12	2	5	-1	16	-7	18	-15	14	-11	4	-13
-16	3	-2	8	13	11	-14	-4	-18	15	-6	17	-5	7	-10	1	-12	9
-2	1	-4	3	-6	5	-8	7	-10	9	-12	11	-14	13	-16	15	-18	17
8	-15	10	-7	18	-9	4	-1	6	-3	14	-13	12	-11	2	-17	16	-5
-4	17	-14	1	-12	13	-10	15	-16	7	-18	5	-6	3	-8	9	-2	11
-7	-11	8	-15	16	18	1	-3	12	-13	2	-9	10	-17	4	-5	14	-6
3	13	-1	11	-10	-16	15	17	-14	5	-4	18	-2	9	-7	6	-8	-12
-15	-9	7	-13	14	12	-3	-11	2	-17	8	-6	4	-5	1	-18	10	16
17	5	15	18	-2	-14	9	13	-7	11	-10	16	-8	6	-3	-12	-1	-4
-11	-18	-5	-9	3	10	-17	-12	4	-6	1	8	15	-16	-13	14	7	2
13	6	17	5	-4	-2	11	9	-8	16	-7	-14	-1	12	18	-10	-3	-15
-9	-16	-11	-6	8	4	-13	-5	1	-12	3	10	7	-18	-17	2	15	14
5	12	13	16	-1	-7	6	18	-15	14	-17	-2	-3	-10	9	-4	11	-8
-6	-14	-9	-12	7	1	-5	-16	3	-18	15	4	17	2	-11	8	-13	10
18	10	12	14	-15	-8	16	6	-17	-2	-13	-3	11	-4	5	-7	9	-1
-12	7	-18	-10	17	15	-2	-14	11	4	-9	1	-16	8	-6	13	-5	3
14	-4	16	2	-11	-17	18	10	-13	-8	5	-15	9	-1	12	-3	6	-7
-10	8	-6	17	9	3	-12	-2	-5	1	-16	7	-18	15	-14	11	-4	13
16	-3	2	-8	-13	-11	14	4	18	-15	6	-17	5	-7	10	-1	12	-9

Slides N. Beldiceanu and H. Simonis

http://www.weltfussball.de/alle_spiele/bundesliga-2010-2011/

Result

J	Scheme	Ref	Trans	Constraint
1	scheme(612,34,18,1,18)	284	absolute_value	symmetric_alldifferent([1..18])*34
2	vector(612)	289	id	global_cardinality([-18.. -1-17,0-0,1..18-17])*1
3	scheme(612,34,18,34,1)	288	id	alldifferent*18
4	repart(612,34,18,17,18)	282	id	alldifferent*306
5	scheme(612,34,18,2,2)	286	id	alldifferent*153
6	scheme(612,34,18,1,18)	284	id	alldifferent*34
7	repart(612,34,18,34,9)	283	sign	alldifferent*306
8	scheme(612,34,18,17,1)	287	absolute_value	alldifferent*36
9	scheme(612,34,18,2,1)	285	absolute_value	alldifferent*306
10	repart(612,34,18,34,9)	283	id	sum_ctr(0)*306
11	repart(612,34,18,34,9)	283	id	sum_cubes_ctr(0)*306
12	scheme(612,34,18,1,18)	284	id	sum_squares_ctr(2109)*34
13	repart(612,34,18,34,9)	283	id	twin*1
14	repart(612,34,18,34,9)	283	id	elements([*i*,-*i*])*1
15	modulo(612,4)	281	id	all_differ_from_at_least_k_pos(152)*1
16	first(9,[1,3,5,7,9,11,13,15,17])	280	id	strictly_increasing*1
17	repart(612,34,18,34,9)	283	id	alldifferent_interval(2)*306
18	scheme(612,34,18,2,1)	285	id	alldifferent_interval(2)*306
19	repart(612,34,18,34,9)	283	sign	sum_ctr(0)*306
20	scheme(612,34,18,1,18)	284	sign	sum_ctr(0)*34
21	repart(612,34,18,34,9)	283	sign	twin*1
22	repart(612,34,18,34,9)	283	absolute_value	twin*1
23	repart(612,34,18,34,9)	283	sign	elements([*i*,-*i*])*1
24	repart(612,34,18,34,9)	283	absolute_value	elements([*i*,*i*])*1
25	first(9,[1,3,5,7,9,11,13,15,17])	280	absolute_value	strictly_increasing*1
26	first(6,[1,4,7,10,13,16])	279	absolute_value	strictly_increasing*1
27	repart(612,34,18,34,9)	283	sign	alldifferent_interval(2)*306
28	scheme(612,34,18,34,1)	288	sign	among_seq(3,[-1])*18

Slides N. Beldiceanu and H. Simonis

Points to remember

- Learning constraint models from positive examples
- Start with **vector** of values
- Group into **regular pattern**
- Find constraint pattern that apply to group elements
- Using ***Constraint Seeker*** for *Global Constraint Catalog*
- Works for **highly structured** problems

Slides N. Beldiceanu and H. Simonis

Partition generators

Structured groups of variables passed to

a conjunction of **identical** constraints

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

Slides N. Beldiceanu and H. Simonis

Partition generators

Structured groups of variables passed to

a conjunction of **identical** constraints

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sum_ctr(34)*4

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sum_ctr(34)*4

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

strictly_decreasing*2
sum_ctr(34)*2

Slides N. Beldiceanu and H. Simonis

Partition generators

Structured groups of variables passed to

a conjunction of **identical** constraints

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Slides N. Beldiceanu and H. Simonis

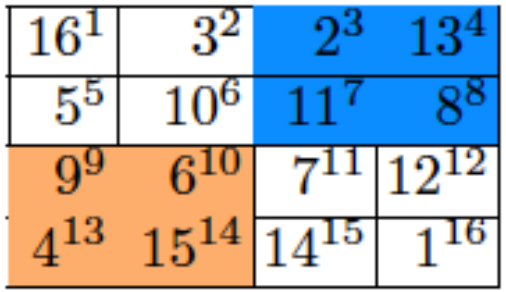
Partition generators

Structured groups of variables passed to

a conjunction of **identical** constraints

sum_ctr(34)*4

surprise

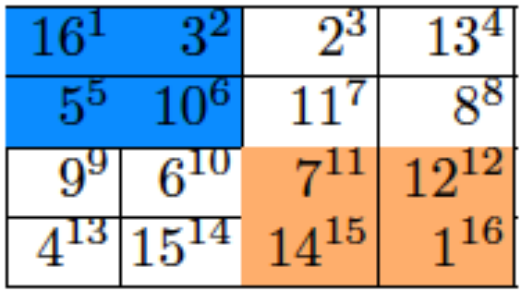


sum_squares_ctr(358)*2

surprise

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1



sum_squares_ctr(390)*2

surprise

Slides N. Beldiceanu and H. Simonis

Partition generators

Structured groups of variables passed to

a conjunction of **identical** constraints

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

Slides N. Beldiceanu and H. Simonis

Partition generators

Structured groups of variables passed to

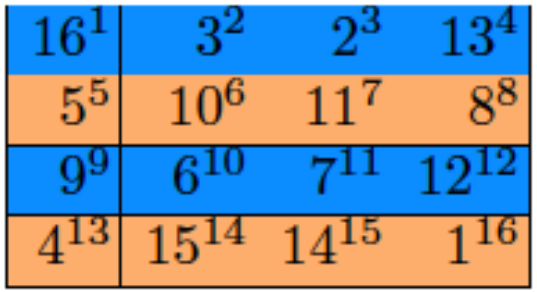
a conjunction of **identical** constraints

sum_ctr(34)*4

surprise

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1



sum_squares_ctr(748)*2

surprise

alldifferent_interval(2)*8

Slides N. Beldiceanu and H. Simonis

Analysis

Clever enumeration of partitions (generate and test)

Per partition :

- clever generation of conjunction of constraints that holds

Multiple examples

- remove constraints that do not hold

Nicely deals with generality / dominance

- relations amongst global constraints builtin catalogue

Generality

Two difficulties

1) $x = y \rightarrow x \geq y$ more general

many solutions can be found

preference for most specific one (Find-S)

2) $x = y$ and $y = z \rightarrow x = z$

therefore $x = y$ and $y = z$ and $x = z$ redundant

equivalent with $x = y$ and $y = z$

Consider now Sudoko ...

Many solutions syntactically different, but semantically equivalent

Generalizations

Generalizations

From propositional logic to first order logic

- Inductive Logic Programming

From ILP to Equation Discovery

From hard to soft constraints

- weighted MAX-SAT
- probabilistic models

Learning preferences

Inductive Logic Programming

Instead of learning propositional formulae, learn first order formulae

Usually (definite) clausal logic

Generalizations of many algorithms exist

Rule learning, decision tree learning

Clausal discovery *[De Raedt MLJ 97, De Raedt AIJ 94]*

- generalizes k-CNF of Valliant to first order case
- enumeration process as for k-CNF with border ...

Clausal Discovery in ILP

train(utrecht, 8, 8, denbosch) ←
train(maastricht, 8, 10, weert) ←
train(utrecht, 9, 8, denbosch) ←
train(maastricht, 9, 10, weert) ←
train(utrecht, 8, 13, eindhoven) ←
train(utrecht, 8, 43, eindhoven) ←
train(utrecht, 9, 13, eindhoven) ←
train(utrecht, 9, 43, eindhoven) ←

train(tilburg, 8, 10, tilburg) ←
train(utrecht, 8, 25, denbosch) ←
train(tilburg, 9, 10, tilburg) ←
train(utrecht, 9, 25, denbosch) ←
train(tilburg, 8, 17, eindhoven) ←
train(tilburg, 8, 47, eindhoven) ←
train(tilburg, 9, 17, eindhoven) ←
train(tilburg, 9, 47, eindhoven) ←

From1 = From2 ← train(From1, Hour1, Min, To), train(From2, Hour2, Min, To)

Inducing constraints that hold in data points
here functional dependencies
[De Raedt 97 MLJ, Flach AIComm 99,
Abdennaher CP 00, Lopez et al ICTAI 10, ...]

Clausal Discovery

Example : one interpretation

{ human(luc), human(lieve), male(luc), female(lieve) }

L : no constants/one variable

Find :

human(X) :- male(X).

human(X) :- female(X).

female(X); male(X) :- human(X)

false :- male(X), female(X).

so this is ako first order 3-CNF

Clausal Discovery

Observe :

if $h_1; \ldots; h_n$:- $b_1, \ldots, b_m$ does not cover e

then there is an answer to

:-$b_1, \ldots, b_m$, not $h_1, \ldots$,not h_n

therefore consider refinements :

$h; h_1; \ldots, h_n$:- $b_1, \ldots, b_m$ and

$h_1; \ldots, h_n$:- $b_1, \ldots, b_m, b$

in all possible ways

Clausal Discovery

Observe :

if $h_1; \ldots; h_n$:- $b_1, \ldots, b_m$ satisfies e

then refinements

$h; h_1; \ldots, h_n$:- $b_1, \ldots, b_m$ and

$h_1; \ldots, h_n$:- $b_1, \ldots, b_m, b$

also satisfy e

Clausal Discovery

Q := { false :- true }

H := {}

while Q is not empty do

delete c from Q

if c covers all p in P *(and H does not entail c)*

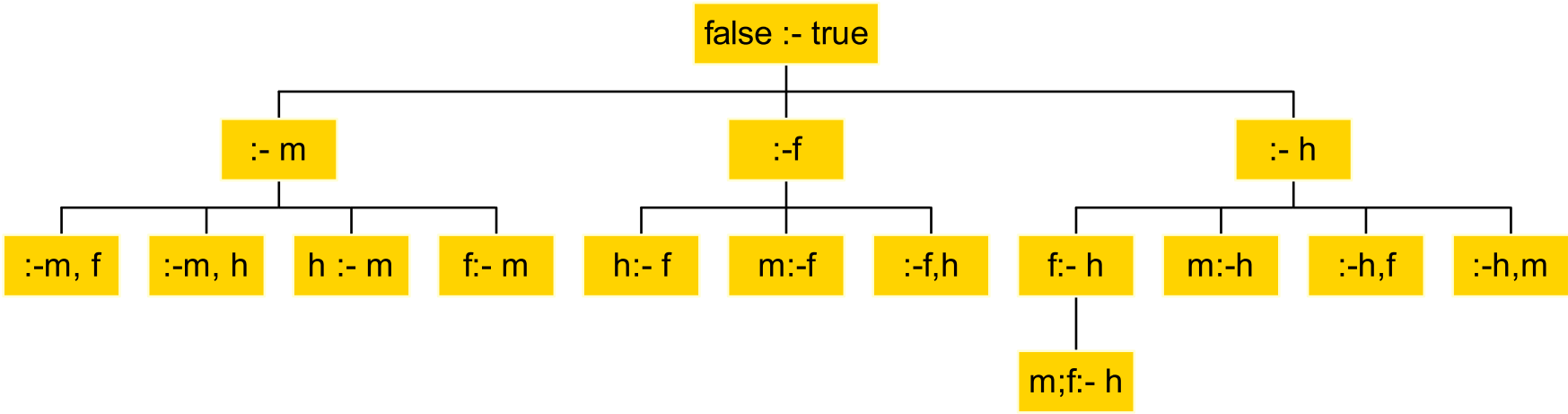
then add c to H

else add all refinements of c to Q

(pruning : generate clauses in L at most once

heuristics)

Clausal Discovery



Map Colouring

- Boolean approach

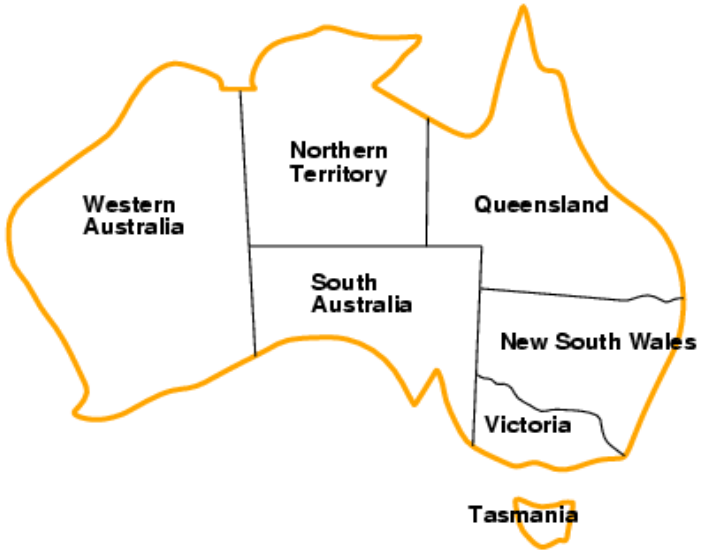
  WA ≠ NT and NT ≠ SA and WA ≠ SA ...

- Relational approach

  X ≠ Y :- next(X,Y).

  but requires next to be given

Map Colouring



- Variables WA, NT, Q, NSW, V, SA, T
- Domains D_i = {red,green,blue}
- Constraints: adjacent regions must have different colors
- e.g., WA ≠ NT, or (WA,NT) in {(red,green),(red,blue),(green,red), (green,blue),

Equation Discovery

Instead of learning clauses, learn equations *[Dzeroski and Todorovski, Langley and Bridewell]*.

As Valiant's algorithm

- generate and test candidate equations, e.g., ax + byz = c
  - fit parameters using regression
- possibly compute values for additional variables (partial derivatives w.r.t. time, etc.)
- include a grammar to specify "legal equations" (bias)

Table 1
Variables used in the NPPc portion of the CASA model

NPPc is the net production of carbon by terrestrial plants at a site
E is the photosynthetic efficiency at a site after factoring various sources of stress
T1 is a temperature stress factor ($0 < T1 < 1$) for cold weather
T2 is a temperature stress factor ($0 < T2 < 1$), nearly Gaussian in form but falling off more quickly at higher temperatures
W is a water stress factor ($0.5 < W < 1$)
topt is the average temperature for the month at which *fas_ndvi* takes on its maximum value at a site
tempc is the average temperature at a site for a given month
eet is the estimated evapotranspiration (water loss due to evaporation and transpiration) at a site
PET is the potential evapotranspiration (water loss due to evaporation and transpiration given an unlimited water supply) at a site
pet_tw_m is a component of potential evapotranspiration that takes into account the latitude, time of year, and days in the month
A is a polynomial function of the annual heat index at a site
ahi is an annual heat index that takes the time of year into account
fas_ndvi is the relative greenness as measured from space
IPAR is the energy intercepted from the sun after factoring in the time of year and days in the month
FPAR_FAS is the fraction of energy intercepted from the sun that is absorbed photosynthetically after factoring in vegetation type
monthly_solar is the average radiation incoming for a given month at a site
SOL_CONV is 0.0864 times the number of days in each month

Ecological Modeling

$$
\begin{aligned}
NPPc &= \max(0, E \cdot IPAR) \\
E &= 0.312 \cdot T1^{1.36} \cdot T2^{0.728} \cdot W^{0} \\
T1 &= 3.65 - 0.992 \cdot topt + 0.137 \cdot topt^2 - 0.00679 \cdot topt^3 + 0.000111 \cdot topt^4 \\
T2 &= 0.818/((1 + \exp(0.0521 \cdot (TDIFF - 10))) \cdot (1 + \exp(0 \cdot (-TDIFF - 10)))) \\
TDIFF &= topt - tempc \\
W &= 0.5 + 0.5 \cdot eet/PET \\
PET &= 1.6 \cdot (10 \cdot \max(tempc, 0)/ahi)^{A} \cdot pet\_tw\_m \\
A &= 0.000000675 \cdot ahi^3 - 0.0000771 \cdot ahi^2 + 0.01792 \cdot ahi + 0.49239 \\
IPAR &= FPAR\_FAS \cdot monthly\_solar \cdot SOL\_CONV \cdot 0.5 \\
FPAR\_FAS &= \min((SR\_FAS - 1.08)/srdiff, 0.95) \\
SR\_FAS &= (1 + fas\_ndvi/750)/(1 - fas\_ndvi/750) \\
SOL\_CONV &= 0.0864 \cdot days\_per\_month
\end{aligned}
$$

Using equation discovery to revise an Earth ecosystem model of the carbon net production

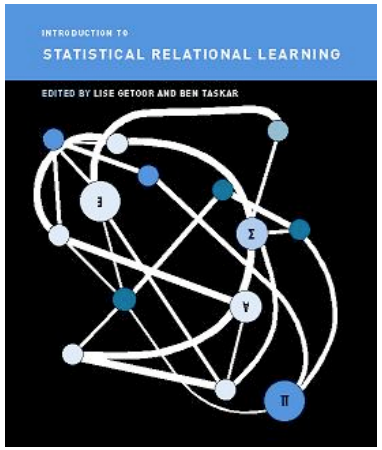
Ljupčo Todorovski [a,*], Sašo Džeroski [a],
Pat Langley [b], Christopher Potter [c]

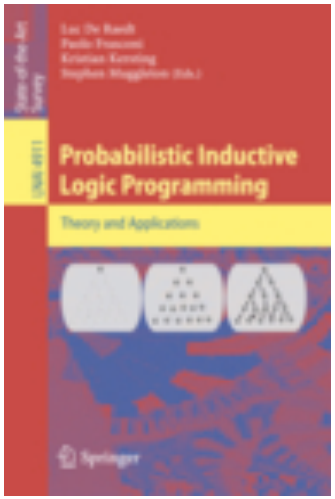
Learning Soft Constraints

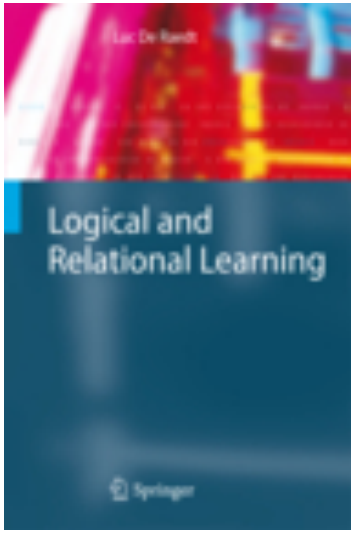
Let us look at weighted MAX-SAT problems

Quite popular today in Statistical Relational Learning

- combining first order logic, machine learning and uncertainty
- One example is Markov Logic, many others exist







Factors and Logic

- Propositional atoms are binary (0-1) variables.
- A joint instantiation of all atoms/variables satisfying a propositional formula is a *model* of that formula.
- If A and B are the only propositions in our language then $A, \neg A \vee B$ has only one model.

A	
-	-
0	0
1	1

*

A	B	
-	-	-
0	0	1
0	1	1
1	0	0
1	1	1

=

A	B	
-	-	-
0	0	0
0	1	0
1	0	0
1	1	1

Slide James Cussens

Generalizing Propositional Logic

- Allow arbitrary non-negative values in the factors.
- Allow variables to have more than 2 values.

A			A	B			A	B	
-	-		-	-	-		-	-	-
0	4	*	0	0	5	=	0	0	20
1	6		0	1	5		0	1	20
			1	0	0		1	0	0
			1	1	7		1	1	42

Dividing by a normalising constant Z defines a probability distribution over full joint instantiations (when $Z > 0$). Here $Z = 20 + 20 + 0 + 42 = 82$.

Slide James Cussens

Weighted Clauses

$\infty : A$ and $2 : \neg A \vee B$

A			A	B			A	B	
-	-		-	-	-		-	-	-
0	0	*	0	0	1	=	0	0	0
1	1		0	1	1		0	1	0
			1	0	exp(-2)		1	0	exp(-2)
			1	1	1		1	1	1

Finding the most probable instantiation (highest weighted model) is the weighted MAX-SAT problem.

e^{-w} where w=weight of clause if clause not satisfied;
weight = 0 otherwise

Slide James Cussens

Weighted Logic

Markov Logic uses weighted (first order logic) clauses to represent a Markov Network

Interesting inference and learning problems

- Compute P(X|Y) ... (CP-techniques can help, weighted model counting)
- Compute most likely state (MAX-SAT)
- Learn parameters (weights of clauses)
  - e.g., using gradient descent on likelihood
- Learn structure and parameters

[Domingos et al], related to [Rossi, Sperduti KR, JETAI etc]

Learning Probabilistic Models

Given

- a space of possible instances X
- an unknown target function P: X → Y Y=[0,1]
- a hypothesis space L containing functions X → Y (graphical models)
- a set of examples E = { (x, _) | x ∈ X }
- a loss function *loss*(h,E) → $\mathbb{R}$

generative

$$\prod_{e \in E} P(e|h)$$

Find h ∈ L that minimizes *loss*(h,E)

maximize likelihood

generative

Parameter Estimation

incomplete data set

A1	A2	A3 (hidden/ latent)	A4	A5	A6
true	true	?	true	false	false
?	true	?	?	false	false
...	...	...	...	...	...
true	false	?	false	true	?

states of some random variables are missing

E.g. medical diagnosis

missing value

Preference learning

Problem with previous approach

- hard to sample examples from probability distribution in CP context; or to give examples with target probability

A hot topic today in ML, many variations exist, cf. *[Furnkranz and Eykemuller, 10, book & tutorial -- videolectures]*

Two main settings

- learning object preferences (model acquisition)
- learning label preferences (portfolio's)

Object Preferences

Given

- a space of possible instances X
- an unknown ranking function r(.), given $O \subseteq X$, rank instances in O
- a hypothesis space L containing ranking functions
- a set of examples $E = \{ (x > y) \mid x,y \in X \}$
- a loss function *loss*(h,E) $\rightarrow \mathbb{R}$

Find $h \in L$ that minimizes *loss*(h,E)

Possible approaches

Explicit relation learning

- Learn a relation Q(x,y) from examples $x < y$
- Determine r(O) as the ordering that is maximally consistent with Q

Learn latent utility function

- an unknown utility function $f: X \rightarrow \mathbb{R}$
- examples only impose constraints on f
  - values of f not known

Label Preferences

Given

- a space of possible instances X
- a set of labels $Y = \{Y_1, \ldots, Y_n\}$
- an unknown target function f(x) = permutation of Y
- a set of examples $E = \{ (x , \{ Y_i > Y_j \})\}$
- a loss function *loss*(h,E) → $\mathbb{R}$

Find h ∈ L that minimizes *loss*(h,E)

Possible approaches

Learn set of relations for each $Y_i > Y_j$

Learn latent utility function for each label Y_i

An unknown utility function $f_i : X \rightarrow \mathbb{R}$

- examples only impose constraints on f_i :
  - values of f not known

Summary

The learning of CSPs is possible, so let's do it

Many settings exist

- data, hypothesis language, active, soft constraints, preference learning, etc

Still we did not touch upon

- bayesian and statistical learning methods

One interesting approach that learns MAX-SAT and MAX-SMT by asking preference questions and using statistical learning techniques

Campigotto, A. Passerini and R. Battiti, Lion 10 workshop

Further reading -- Encyclopedia of Machine Learning
-- Mitchell, Machine Learning,

References [Part II]

Tom Mitchell, Machine Learning, Mc GrawHill, 1997.

Encyclopedia of Machine Learning, Springer, 2010.

Johannes Fürnkranz and Eyke Hüllermeier, Preference Learning, in: Encyclopedia of Machine Learning, Springer-Verlag, 2010. Also, edited book and videolectures !

C. Bessiere, et al. Learning constraint networks IJCAI 07, IJCAI 13, AAAI 14

Campigotto, A. Passerini and R. Battiti, Lion 10 workshop

Alessandro Biso, Francesca Rossi, Alessandro Sperduti: Experimental Results on Learning Soft Constraints. KR 2000: 435-444 and later papers

Pedro Domingos, Daniel Lowd: Markov Logic: An Interface Layer for Artificial Intelligence Morgan & Claypool Publishers 2009

Dzeroski, S. and Todorovski, L. (1995) Discovering dynamics: From inductive logic programming to machine discovery. Journal of Intelligent Information Systems, 4: 89-108. and later papers

Luc De Raedt, Luc Dehaspe: Clausal Discovery. Machine Learning 26(2-3): 99-146 (1997)

Luc De Raedt, Logical and Relational Learning, Springer, 2008.

References [Part II]

Dana Angluin: Queries and Concept Learning. Machine Learning 2(4): 319-342(1987)

Dana Angluin, Michael Frazier, Leonard Pitt: Learning Conjunctions of Horn Clauses. Machine Learning 9: 147-164 (1992)

Michael Frazier, Leonard Pitt: Learning From Entailment: An Application to Propositional Horn Sentences. ICML 1993: 120-127

Peter A. Flach, Iztok Savnik: Database Dependency Discovery: A Machine Learning Approach. AI Commun. 12(3): 139-160 (1999)

Slim Abdennadher, Christophe Rigotti: Automatic Generation of Propagation Rules for Finite Domains. CP 2000: 18-34

Arnaud Lallouet, Matthieu Lopez, Lionel Martin, Christel Vrain: On Learning Constraint Problems. ICTAI (1) 2010: 45-52

Thank you